

EXERCÍCIO HERANÇA

The background features a dark, moody scene of a person's hand holding a glowing blue wireframe sphere above an open laptop. A large, solid pink circle is positioned behind the word 'HERANÇA' in the title, making it stand out. In the top right corner, there is a faint, stylized icon of a gear with circuit lines extending from it. The bottom left corner shows a faint, light-colored network diagram with nodes and connecting lines. The overall aesthetic is technological and modern.

Árvore Binária



01

Classe base

Implemente uma classe que representa uma árvore de busca binária, sem balanceamento.

02

Implementação

Essa classe deve ser implementada usando um array de Strings, no formato de um heap.

03

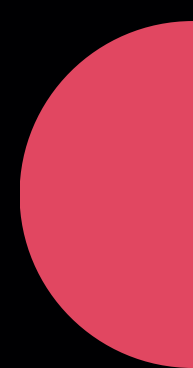
Heap

Posição 0 do array é a raiz. Filho esquerdo do nó i está na posição $(2 * i + 1)$. Filho direito na posição $(2 * i + 2)$.

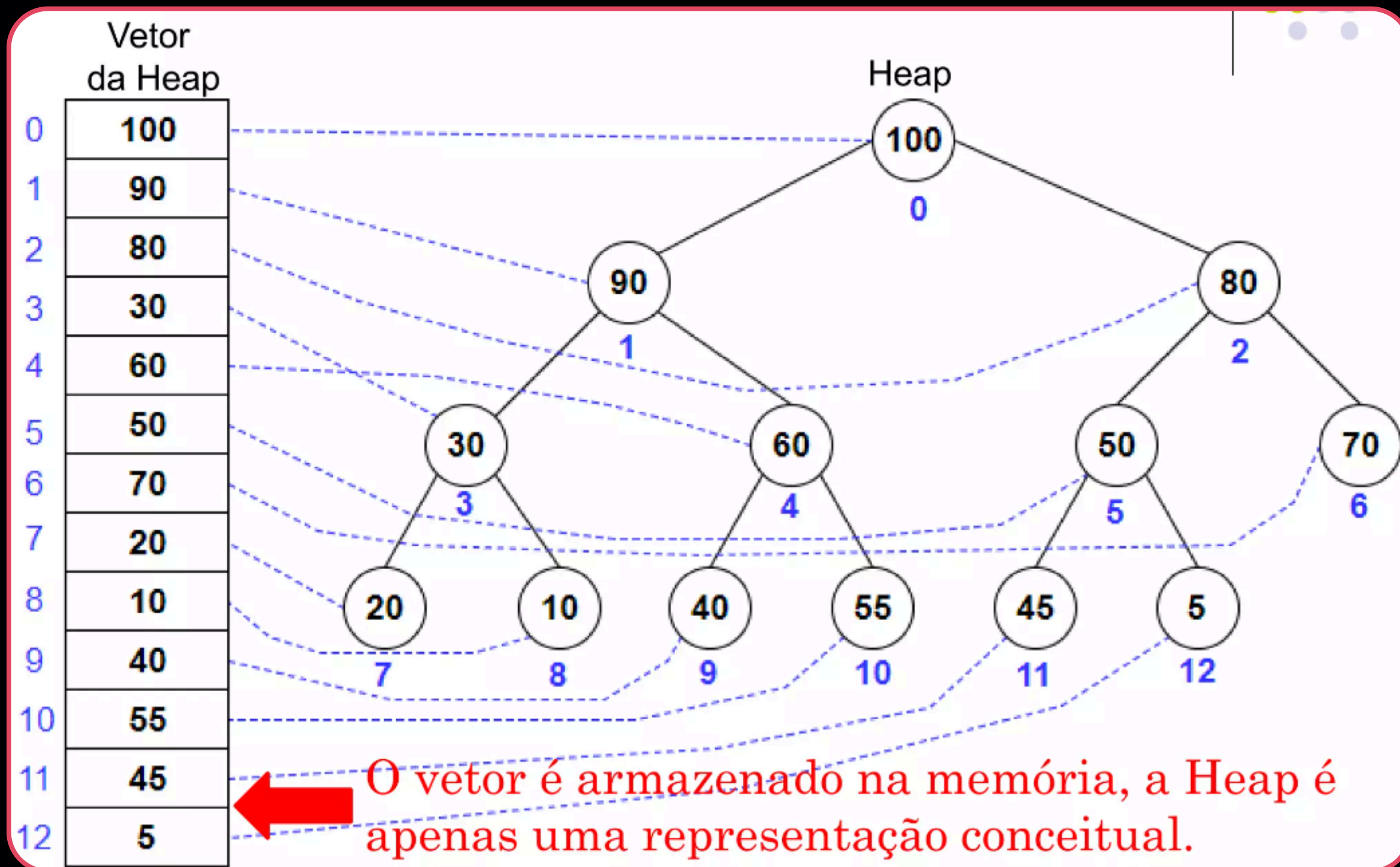
04

Interface pública

Inserir, remover, procurar, tamanho, toString.



Heap



Obrigatório

Method Summary

All Methods

Instance Methods

Concrete Methods

Modifier and Type	Method	Description
boolean	<code>find(String[?] v)</code>	Verifica se o elemento está presente na árvore.
void	<code>insert(String[?] value)</code>	Insere um string na árvore
int	<code>len()</code>	Reorna o número de elementos presentes na árvore
boolean	<code>remove(String[?] v)</code>	Remove um elemento da árvore.
<code>String[?]</code>	<code>toString()</code>	Retorna um string no formato de um grafo da ferramenta graphviz.

Obrigatório

Constructor Summary

Constructors

Constructor	Description
<code>ArvBin(int len)</code>	Construtor gera um aárvore vazia.

Method Summary

All Methods

Instance Methods

Concrete Methods

Modifier and Type	Method	Description
boolean	<code>find(String[?] v)</code>	Verifica se o elemento está presente na árvore.
void	<code>insert(String[?] value)</code>	Insere um string na árvore
int	<code>len()</code>	Reorna o número de elementos presentes na árvore
boolean	<code>remove(String[?] v)</code>	Remove um elemento da árvore.
<code>String[?]</code>	<code>toString()</code>	Retorna um string no formato de um grafo da ferramenta graphviz.

Árvore Binária Balanceada



01

Subclasses

Implemente uma subclasse que representa uma árvore de busca binária, com balanceamento perfeito.

03

Heap

Serão necessários métodos protegidos na superclasse para manipular o heap.

02

Subclasses

Implemente uma classe que representa uma árvore AVL.

04

Interface pública

Inserir, remover, procurar, tamanho, toString.

Subclasses

Class ArvAVL

java.lang.Object 
ArvBin
ArvAVL

```
public class ArvAVL  
extends ArvBin
```

Class ArvBal

java.lang.Object 
ArvBin
ArvBal

```
public class ArvBal  
extends ArvBin
```

Protegidos - superclasse

All Methods	Instance Methods	Concrete Methods
Modifier and Type	Method	Description
protected int	<code>countNodes(int i)</code>	Conta o número de nós em um subárvore.
boolean	<code>find(String[Ⓔ] v)</code>	Verifica se o elemento está presente na árvore.
protected String [Ⓔ]	<code>getNode(int i)</code>	Retorna o valor de um dado nó.
void	<code>insert(String[Ⓔ] value)</code>	Insere um string na árvore
protected boolean	<code>isBalance()</code>	Verifica se a árvore está balanceada, de acordo com o critério de cada tipo de árvore.
int	<code>len()</code>	Reorna o número de elementos presentes na árvore
protected int	<code>nodeLeft(int i)</code>	Indica qual é o número do filho à esquerda da posição i.
protected int	<code>nodeRight(int i)</code>	Indica qual é o número do filho à direita da posição i.
boolean	<code>remove(String[Ⓔ] v)</code>	Remove um elemento da árvore.
protected void	<code>setNode(int i, String[Ⓔ] v)</code>	Atribui um valor para um determinado nó.
<code>String[Ⓔ]</code>	<code>toString()</code>	Retorna um string no formato de um grafo da ferramenta graphviz.

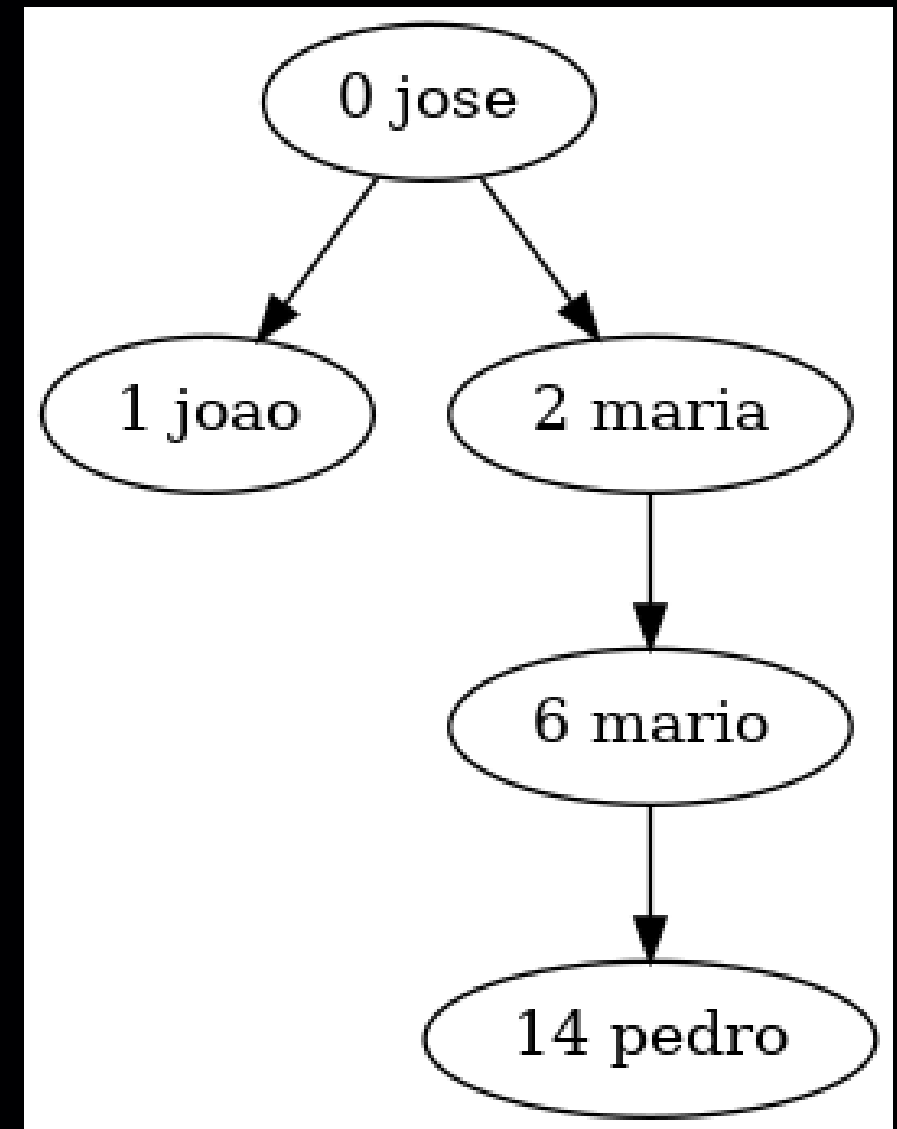
toString()

```
digraph {  
  "0 jose" -> "1 joao"  
  "0 jose" -> "2 maria"  
  "2 maria" -> "6 mario"  
  "6 mario" -> "14 pedro"  
}
```

toString()

```
digraph {  
  "0 jose" -> "1 joao"  
  "0 jose" -> "2 maria"  
  "2 maria" -> "6 mario"  
  "6 mario" -> "14 pedro"  
}
```

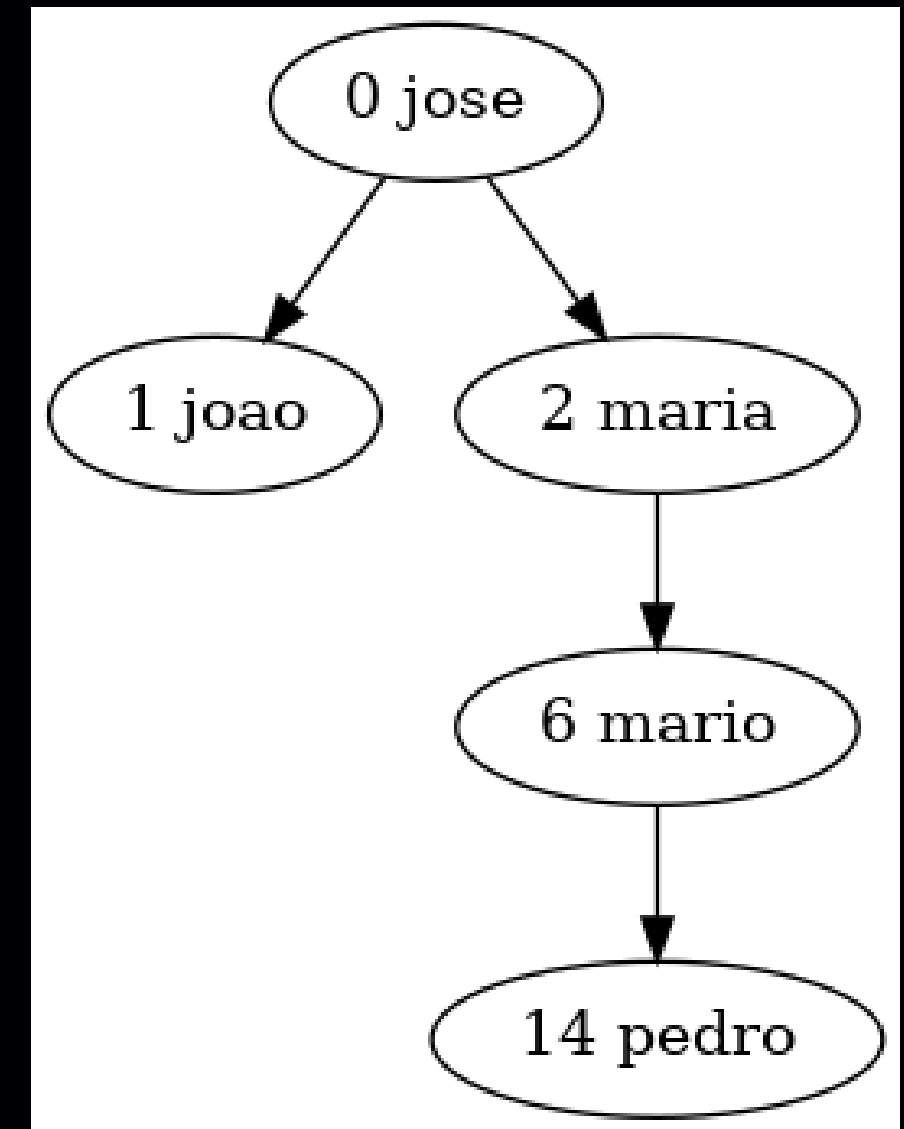
graphviz



toString()

```
digraph {  
  "0 jose" -> "1 joao"  
  "0 jose" -> "2 maria"  
  "2 maria" -> "6 mario"  
  "6 mario" -> "14 pedro"  
}
```

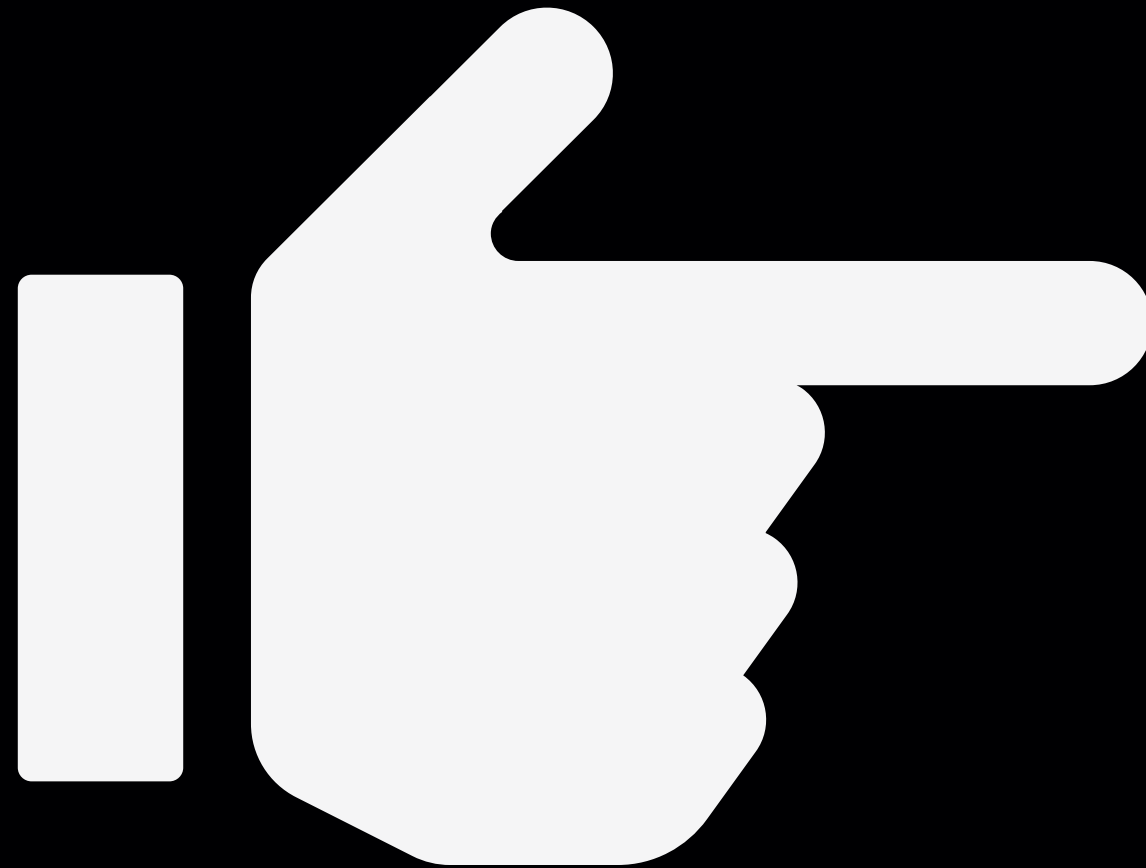
graphviz



>> dot -Tpng -o arq.png arq.dot

Entrada/Saída

```
i jose  
i joao  
i maria  
i jose  
d joao  
i mario
```



**toString() para
cada uma das
árvores**

Dicas

- Cada vez que um elemento é inserido ou removido, é preciso verificar o balanceamento da árvore
- Se ela estiver desbalanceado, aplica algoritmo de rebalanceamento
- APB: número de elementos das sub-árvores difere no máximo em 1 unidade
- AVL: altura das sub-árvores difere no máximo em 1 unidade

Dicas

- APB: “jogar fora” a árvore atual e re-inserir os elementos novamente, de forma que ela fique perfeitamente balanceada
- AVL: fazer as rotações necessárias